Spojrzenie na logikę rozmytą

Wyraźna czy rozmyta logika?

Logika zajmuje się prawdą i fałszem. Zdanie może być raz prawdziwe, a innym razem fałszywe. „Jabłko to czerwony owoc" to taka propozycja. Jeśli trzymasz zielone jabłko Granny Smith, twierdzenie, że jabłko jest czerwonym owocem, jest fałszywe. Z drugiej strony, jeśli twoje jabłko jest czerwonej, pysznej odmiany, jest to czerwony owoc i propozycja, o której mowa, jest prawdziwa. Jeśli zdanie jest prawdziwe, ma wartość prawdziwości 1; jeśli jest fałszywe, jego wartość logiczna wynosi 0. Są to jedyne możliwe wartości logiczne. Propozycje można łączyć w celu generowania innych propozycji za pomocą operacji logicznych. Kiedy mówisz, że dzisiaj będzie padać lub że będziesz mieć dziś piknik na świeżym powietrzu, z całą pewnością składasz oświadczenia. Oczywiście Twoje wypowiedzi w tym przypadku mogą być prawdziwe lub fałszywe. Prawda w twoich wypowiedziach może wynosić tylko 1 lub 0. Twoje wypowiedzi można wtedy uznać za szorstkie. Z drugiej strony są stwierdzenia, których nie można złożyć z taką pewnością. Możesz powiedzieć, że myślisz, że dzisiaj będzie padać. Jeśli będziesz naciskany dalej, możesz z pewną dozą pewności stwierdzić w swoim oświadczeniu, że dzisiaj będzie padać. Twój poziom pewności wynosi jednak około 0,8, a nie 1. Do modelowania tego typu sytuacji opracowano logikę rozmytą. Logika rozmyta zajmuje się zdaniami, które mogą być prawdziwe do pewnego stopnia - gdzieś od 0 do 1. Dlatego wartość prawdziwości zdania wskazuje stopień pewności, co do którego to zdanie jest prawdziwe. Stopień pewności brzmi jak prawdopodobieństwo (być może prawdopodobieństwo subiektywne), ale to nie to samo. Prawdopodobieństwo zdarzeń wzajemnie wykluczających się nie może sumować się do więcej niż 1, ale ich rozmyte wartości mogą. Załóżmy, że prawdopodobieństwo, że filiżanka kawy jest gorąca, wynosi 0,8, a prawdopodobieństwo, że filiżanka kawy jest zimna, wynosi 0,2. Te prawdopodobieństwa muszą się sumować do 1,0. Wartości rozmyte nie muszą być sumowane do 1,0. Prawda twierdzenia, że filiżanka kawy jest gorąca, wynosi 0,8. Prawda twierdzenia, że filiżanka kawy jest zimna, może wynosić 0,5. Nie ma ograniczeń co do tego, do czego muszą się składać te wartości prawdy.

Zbiory rozmyte

Logika rozmyta jest najlepiej rozumiana w kontekście przynależności do zbioru. Załóżmy, że zbierasz zestaw deszczowych dni. Czy umieściłbyś dzisiaj w zbiorze? Kiedy masz do czynienia tylko z ostrymi stwierdzeniami, które są albo prawdziwe, albo fałszywe, twoje włączenie dnia dzisiejszego do zestawu deszczowych dni opiera się na pewności. Mając do czynienia z logiką rozmytą, do zestawu dni deszczowych należy doliczyć dzień dzisiejszy za pomocą uporządkowanej pary, takiej jak (dzisiaj, 0.8). Pierwszy członek w tak uporządkowanej parze jest kandydatem do włączenia do zbioru, a drugi członek to wartość z zakresu od 0 do 1 włącznie, nazywana stopniem przynależności do zbioru. Włączenie stopnia przynależności do zbioru ułatwia programistom wymyślenie teorii mnogości opartej na logice rozmytej, tak jak rozwija się regularną teorię mnogości. Zbiory rozmyte to zbiory, w których członkowie są przedstawiani jako uporządkowane pary zawierające informacje o stopniu przynależności. Tradycyjny zbiór, powiedzmy, k elementów, jest szczególnym przypadkiem zbioru rozmytego, w którym każdy z tych k elementów ma stopień przynależności 1, a każdy inny element zbioru uniwersalnego ma stopień przynależności 0, dla którego powód, dla którego nie zadajesz sobie trudu, aby go wymienić.

Operacje na zestawach rozmytych

Zwykłe operacje, które możesz wykonać na zwykłych zestawach, to suma, w której bierzesz wszystkie elementy, które są w jednym lub drugim zestawie; i skrzyżowanie, w którym bierzesz elementy znajdujące się w obu zestawach. W przypadku zbiorów rozmytych branie sumy polega na znalezieniu stopnia przynależności, jaki element powinien mieć w nowym zbiorze rozmytym, który jest sumą

dwóch zbiorów rozmytych. Jeśli a, b, c i d są takie, że ich stopnie przynależności do zbioru rozmytego A wynoszą odpowiednio 0,9, 0,4, 0,5 i 0, to zbiór rozmyty A jest określony przez wektor dopasowania (0,9, 0,4, 0,5 , 0). Składniki tego wektora dopasowania nazywane są wartościami dopasowania a, b, c i d.

Unia zbiorów rozmytych

Rozważ połączenie dwóch tradycyjnych zbiorów i elementu, który należy tylko do jednego z tych zbiorów. Wcześniej widziałeś, że jeśli traktujesz te zbiory jako zbiory rozmyte, ten element ma stopień przynależności 1 w jednym przypadku i 0 w drugim, ponieważ należy do jednego, a nie do drugiego. A jednak zamierzasz umieścić ten element w unii. Kryterium, którego używasz w tej akcji, ma związek ze stopniami członkostwa. Musisz przyjrzeć się dwóm stopniom przynależności, a mianowicie 0 i 1, i wybrać wyższą wartość z nich, a mianowicie 1. Innymi słowy, czego chcesz dla stopnia przynależności elementu wymienionego w unii dwóch zbiorów rozmytych, to maksymalna wartość jego stopni przynależności w obrębie dwóch zbiorów rozmytych tworzących unię. Jeśli a, b, c i d mają odpowiednie stopnie przynależności do zbiorów rozmytych A, B jako A = (0,9, 0,4, 0,5, 0) i B = (0,7, 0,6, 0,3, 0,8), to A [kubek]B = (0,9, 0,6, 0,5, 0,8).

Przecięcie i dopełnienie dwóch zbiorów rozmytych

Analogicznie, stopień przynależności elementu w przecięciu dwóch zbiorów rozmytych jest minimalną lub mniejszą wartością jego stopnia przynależności indywidualnie w dwóch zbiorach tworzących przecięcie. Np. jeśli dzisiaj jest 0,8 stopnia przynależności do zbioru dni deszczowych i 0,5 stopnia przynależności do zbioru dni wykonania pracy, to dzień dzisiejszy należy do zbioru dni deszczowych, w których praca jest wykonywana w stopniu 0,5, mniejsza o 0,5 i 0,8. Przypomnijmy zbiory rozmyte A i B z poprzedniego przykładu. A = (0,9, 0,4, 0,5, 0) i B = (0,7, 0,6, 0,3, 0,8). A[∩]B, który jest przecięciem zbiorów rozmytych A i B, otrzymuje się biorąc w każdym składniku mniejszą z wartości znajdujących się w tym składniku w A i B. Zatem A[∩]B = (0,7, 0,4, 0,3, 0). Idea zbioru uniwersalnego jest zawarta w radzeniu sobie ze zbiorami tradycyjnymi. Na przykład, jeśli mówisz o zbiorze osób w związku małżeńskim, zbiorem uniwersalnym jest zbiór wszystkich osób. Każdy inny zbiór, który rozważasz w tym kontekście, jest podzbiorem zbioru uniwersalnego. Poruszamy kwestię zbioru uniwersalnego, ponieważ przy dopełnianiu zbioru tradycyjnego A trzeba wstawić każdy element zbioru uniwersalnego, którego nie ma w zbiorze A. Dopełnienie zbioru rozmytego uzyskuje się jednak w następujący sposób . W przypadku zbiorów rozmytych, jeżeli stopień przynależności danego elementu wynosi 0,8, to ten element nie znajduje się w tym zbiorze w stopniu 1,0 – 0,8 = 0,2. Można więc ustawić stopień przynależności w zbiorze rozmytym dopełnienia do dopełnienia w odniesieniu do 1. Jeśli wrócimy do scenariusza, w którym w zbiorze dni deszczowych wrócimy do stopnia 0,8, to dzisiaj w zestawie dni deszczowych musi być stopień przynależności 0,2 stopnia. zestaw dni bezdeszczowych lub pogodnych. Kontynuując nasz przykład zbiorów rozmytych A i B, oznaczając dopełnienie A przez A’, mamy A’ = (0,1, 0,6, 0,5, 1) i B’ = (0,3, 0,4, 0,7, 0,2). Zauważ, że A' [∩] B' = (0,3, 0,6, 0,7, 1), który jest również uzupełnieniem A [duże] B. Możesz w podobny sposób zweryfikować, że dopełnienie A [∪] B jest takie samo jak A’[∩] B’. Ponadto A [∩] A' = (0,9, 0,6, 0,5, 1) i A [∩] A' = (0,1, 0,4, 0,5, 0), co nie jest wektorem samych zer, jak miałoby to miejsce w konwencjonalnych zestawach. W rzeczywistości A i A’ będą równe w tym sensie, że ich wektory dopasowania są takie same, jeśli każdy składnik w wektorze dopasowania jest równy 0,5.

Zastosowania logiki rozmytej

Zastosowania zbiorów rozmytych i logiki rozmytej można znaleźć w wielu dziedzinach, w tym w sztucznej inteligencji, inżynierii, informatyce, badaniach operacyjnych, robotyce i rozpoznawaniu wzorców. Pola te są również gotowe do zastosowań w sieciach neuronowych. Wydaje się więc

naturalne, że rozmycie powinno być wprowadzone do samych sieci neuronowych. W każdym obszarze, w którym ludzie muszą sobie pozwolić na podejmowanie decyzji, zbiory rozmyte mogą znaleźć swoje miejsce, ponieważ informacje, na których mają być oparte decyzje, mogą nie zawsze być kompletne, a wiarygodność rzekomych wartości podstawowych parametrów nie zawsze jest pewna.

Przykłady logiki rozmytej

Powiedzmy, że w określonym czasie trzeba wykonać pięć zadań, a każde zadanie wymaga jednej dedykowanej do niego osoby. Załóżmy, że jest sześć osób zdolnych do wykonania tych zadań. Ponieważ masz więcej niż wystarczającą liczbę osób, nie ma problemu z zaplanowaniem tej pracy i jej wykonaniem. Oczywiście, kto zostanie przydzielony do którego zadania, zależy od pewnego kryterium, np. całkowitego czasu realizacji, od którego można dokonać pewnej optymalizacji. Ale załóżmy, że te sześć osób niekoniecznie jest dostępnych w określonym czasie, o którym mowa. Nagle równanie staje się mniej wyraziste. Ocenia się dostępność ludzi. Oto przykład problemu przypisania, w którym można użyć zbiorów rozmytych.

Aplikacje komercyjne

Obecnie istnieje wiele komercyjnych zastosowań logiki rozmytej. Oto kilka przykładów:

• Metro w Sendai w Japonii używa rozmytego kontrolera do sterowania wagonem metra. Ten kontroler przewyższył ludzkie i konwencjonalne kontrolery, zapewniając płynną jazdę pasażerom w każdym terenie i warunkach zewnętrznych.

• Aparaty i kamery wykorzystują logikę rozmytą do regulacji mechanizmów autofokusa i eliminowania drgań spowodowanych drżeniem ręki.

• Niektóre samochody wykorzystują logikę rozmytą do różnych zastosowań sterowania. Nissan posiada patenty na układy hamulcowe z logiką rozmytą, sterowanie skrzynią biegów i wtryskiwacze paliwa. GM stosuje w swoich pojazdach Saturn system rozmytej transmisji.

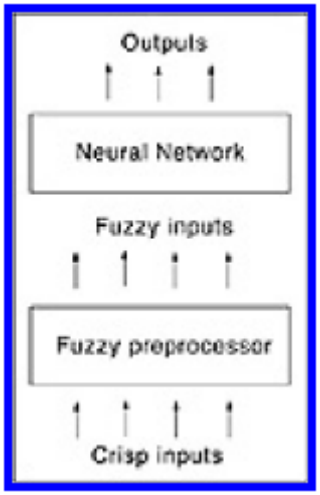
• Firma FuziWare opracowała i opatentowała rozmyty arkusz kalkulacyjny o nazwie FuziCalc, który umożliwia użytkownikom uwzględnienie nieostrości w swoich danych.

• Opracowano aplikacje do wyszukiwania i dopasowywania obrazów dla określonych obszarów zainteresowania pikseli. Avian Systems posiada pakiet oprogramowania o nazwie FullPixelSearch.

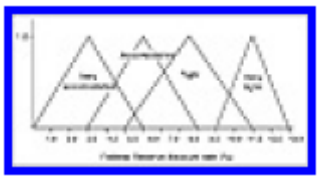
• Narzędzie do tworzenia wykresów i badań giełdowych o nazwie SuperCharts firmy Omega Research wykorzystuje logikę rozmytą w jednym ze swoich modułów w celu określenia, czy rynek jest zwyżkowy, niedźwiedzi czy neutralny.

Rozmycie w sieciach neuronowych

Istnieje wiele sposobów wykorzystania logiki rozmytej w sieciach neuronowych. Być może najprostszym sposobem jest użycie funkcji fuzzifier do wstępnego lub końcowego przetwarzania danych dla sieci neuronowej. Jest to pokazane na rysunku, gdzie sieć neuronowa ma fuzzifier przetwarzania wstępnego, który konwertuje dane na dane rozmyte do zastosowania w sieci neuronowej.



Zbudujmy prosty fuzzifier w oparciu o aplikację do przewidywania kierunku giełdy. Załóżmy, że chcesz rozmyć jeden zestaw danych wykorzystywanych w sieci, politykę fiskalną Rezerwy Federalnej, w jednej z czterech nieostrych kategorii: bardzo akomodacyjna, akomodacyjna, ciasna lub bardzo ciasna. Załóżmy, że surowe dane, które musimy zmyć, to stopa dyskontowa i stopa procentowa kontrolowane przez Rezerwę Federalną w celu ustalenia polityki fiskalnej. Otóż niska stopa dyskontowa zwykle wskazuje na luźną politykę fiskalną, ale zależy to nie tylko od obserwatora, ale także od klimatu politycznego. Istnieje prawdopodobieństwo, że dla danej stopy dyskontowej znajdziesz dwie osoby, które oferują różne kategorie polityki fiskalnej Fed. Dlatego właściwe jest rozmycie danych, tak aby dane, które przedstawiamy sieci neuronowej, były takie, jak zobaczyłby obserwator. Rysunek 2 przedstawia rozmyte kategorie dla różnych stóp procentowych. Zwróć uwagę, że kategoria tight ma największy zasięg. Na dowolnym poziomie stóp procentowych możesz mieć jedną lub kilka możliwych kategorii. Jeśli na wykresie występuje tylko jedna stopa procentowa, oznacza to, że członkostwo w tym zbiorze rozmytym wynosi 1,0. Jeśli masz trzy możliwe zestawy rozmyte, wymagane jest, aby suma członkostwa wynosiła 1,0. Przy oprocentowaniu 8% masz szansę na znalezienie go w kategorii ciasnej lub kategorii akomodacyjnej. Aby znaleźć prawdopodobieństwo procentowe z wykresu, weź wysokość każdej krzywej przy danej stopie procentowej i znormalizuj ją do długości jednej jednostki. Przy 8%, kategoria ciasna ma około 0,8 jednostki wysokości, a akomodacyjna ma około 0,3 jednostki wysokości. Suma wynosi około 1,1 jednostki, a prawdopodobieństwo, że wartość będzie ciasna wynosi wtedy 0,8/1,1 = 0,73, podczas gdy prawdopodobieństwo wartości bycia akomodacyjnym wynosi 0,27.



Kod dla Fuzzifier

Rozwińmy kod w C++, aby stworzyć prosty fuzzifier. Klasa zwana kategorią jest zdefiniowana w listingu 1. Ta klasa zawiera dane, które musimy zdefiniować, kategorie na rysunku 2. Istnieją trzy prywatne elementy danych o nazwach lowval, midval i highval. Reprezentują one wartości na wykresie, które definiują trójkąt kategorii. W kategorii tight lowval wynosi 5,0, midval to 8,5, a highval to 12,0. Klasa kategorii umożliwia utworzenie wystąpienia obiektu kategorii i przypisanie do niego parametrów w celu jego zdefiniowania. Istnieje również ciąg o nazwie name, który identyfikuje kategorię, np. "obcisły." Różne funkcje członkowskie są używane do łączenia się z członkami prywatnych danych.

Istnieje na przykład setval(), która pozwala ustawić wartość trzech parametrów, podczas gdy gethighval() zwraca wartość parametru highval. Funkcja getshare() zwraca względną wartość członkostwa w kategorii podanej na wejściu. W omawianym wcześniej przykładzie, gdzie stopa dyskontowa Fed wynosi 8,0, a kategoria wąska jest zdefiniowana zgodnie z wykresem na rysunku 3.2, metoda getshare() zwróciłaby 0,8. Zauważ, że nie jest to jeszcze znormalizowane. Idąc za tym przykładem, wartość getshare() z kategorii akomodacji również zostałaby użyta do określenia wag członkostwa. Wagi te określają prawdopodobieństwo w danej kategorii. Generator liczb losowych służy do definiowania wartości używanej do wybierania kategorii rozmytej na podstawie zdefiniowanych prawdopodobieństw.

Listing 1 fuzzfier.h

```
// fuzzfier.h V. Rao, H. Rao

// program to fuzzify data

class category

{

private:

char name[30];

float lowval,highval,midval;

public:

category(){};

void setname(char *);

char * getname();

void setval(float&,float&,float&);

float getlowval();

float getmidval();

float gethighval();

float getshare(const float&);

~category(){};

};

int randnum(int);
```

Przyjrzyjmy się plikowi implementacji z listingu 3.2.

Listing 3.2 fuzzfier.cpp

```
// fuzzfier.cpp V. Rao, H. Rao

// program to fuzzify data

#include <iostream.h>
```

```
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <fuzzfier.h>
void category::setname(char *n)
{ strcpy(name,n);
}
char * category::getname()
{ return name;
}
void category::setval(float &h, float &m, float &l)
{ highval=h;
midval=m;
lowval=l;
}
float category::getlowval()
{
return lowval;
}
float category::getmidval()
{
return midval;
}
float category::gethighval()
{
return highval;
}
float category::getshare(const float & input)
{
// this member function returns the relative membership
// of an input in a category, with a maximum of 1.0
```

```
float output;
float midlow, highmid;
midlow=midval-lowval;
highmid=highval-midval;
// if outside the range, then output=0
if ((input <= lowval) || (input >= highval))
output=0;
else
{
if (input > midval)
output=(highval-input)/highmid;
else
if (input==midval)
output=1.0;
else
output=(input-lowval)/midlow;
}
return output;
}
int randomnum(int maxval)
{
// random number generator
// will return an integer up to maxval
srand ((unsigned)time(NULL));
return rand() % maxval;
}
void main()
{
// a fuzzifier program that takes category information:
// lowval, midval and highval and category name
// and fuzzifies an input based on
```

```
// the total number of categories and the membership
// in each category
int i=0,j=0,numcat=0,randnum;
float l,m,h, inval=1.0;
char input[30]=" ";
category * ptr[10];
float relprob[10];
float total=0, runtotal=0;
//input the category information; terminate with `done';
while (1)
{
cout << "\nPlease type in a category name, e.g. Cool\n";
cout << "Enter one word without spaces\n";
cout << "When you are done, type `done' :\n\n";
ptr[i]= new category;
cin >> input;
if ((input[0]=='d' && input[1]=='o' &&
input[2]=='n' && input[3]=='e')) break;
ptr[i]->setname(input);
cout << "\nType in the lowval, midval and highval\n";
cout << "for each category, separated by spaces\n";
cout << " e.g. 1.0 3.0 5.0 :\n\n";
cin >> l >> m >> h;
ptr[i]->setval(h,m,l);
i++;
}
numcat=i; // number of categories
// Categories set up: Now input the data to fuzzify
cout <<"\n\n";
cout << "====================================\n";
cout << "==Fuzzifier is ready for data==\n";
```

```
cout << "==================================\n";
while (1)
{
cout << "\ninput a data value, type 0 to terminate\n";
cin >> inval;
if (inval == 0) break;
// calculate relative probabilities of
// input being in each category
total=0;
for (j=0;j<numcat;j++)
{
relprob[j]=100*ptr[j]->getshare(inval);
total+=relprob[j];
}
if (total==0)
{
cout << "data out of range\n";
exit(1);
}
randnum=randomnum((int)total);
j=0;
runtotal=relprob[0];
while ((runtotal<randnum)&&(j<numcat))
{
j++;
runtotal += relprob[j];
}
cout << "\nOutput fuzzy category is ==> " <<
ptr[j]->getname()<<"<== \n";
cout <<"category\t"<<"membership\n";
cout <<"----------\n";
```

```
for (j=0;j<numcat;j++)

{

cout << ptr[j]->getname()<<"\t\t"<<

(relprob[j]/total) <<"\n";

}

}

cout << "\n\nAll done. Have a fuzzy day !\n";

}
```

Ten program najpierw konfiguruje wszystkie zdefiniowane przez Ciebie kategorie. Może to być przykład, który wybieramy lub dowolny przykład, który możesz wymyślić. Po zdefiniowaniu kategorii możesz zacząć wprowadzać dane, które mają być rozmyte. Gdy wprowadzasz dane, widzisz, że w grę wchodzi aspekt prawdopodobieństwa. Jeśli wpiszesz tę samą wartość dwa razy, możesz otrzymać różne kategorie! Wkrótce zobaczysz przykładowe dane wyjściowe, ale najpierw uwagę techniczną dotyczącą konfiguracji ważonych prawdopodobieństw. Najlepszym sposobem na wyjaśnienie tego jest przykład. Załóżmy, że zdefiniowałeś trzy kategorie, A, B i C. Załóżmy, że kategoria A ma przynależność względną 0,8, kategoria B 0,4, a kategoria C 0,2. W programie te liczby są najpierw mnożone przez 100, więc otrzymujesz A=80, B=40 i C=20. Teraz są one przechowywane w wektorze z indeksem j zainicjowanym, aby wskazywał na pierwszą kategorię. Załóżmy, że te trzy liczby reprezentują trzy sąsiadujące ze sobą pojemniki liczbowe, które są ze sobą połączone. Teraz wybierz losową liczbę do indeksowania w koszu, która ma maksymalną wartość (80+40+20). Jeśli liczba wynosi 100, to jest większa niż 80 i mniejsza niż (80+40), trafiasz do drugiego przedziału, który reprezentuje B. Czy ten schemat daje ważone prawdopodobieństwa? Tak, ponieważ wielkość kosza (przy równomiernym rozłożeniu w nim losowych indeksów) określa prawdopodobieństwo wpadnięcia do kosza. Dlatego prawdopodobieństwo wpadnięcia do kosza A wynosi 80/(80+40+20). Przykładowe wyjście z programu pokazano poniżej. Nasze dane wejściowe są pisane kursywą; dane wyjściowe komputera nie są. W tym przykładzie wprowadzono kategorie zdefiniowane przez wykres na rysunku 3.2. Po skonfigurowaniu kategorii pierwszy wpis danych 4.0 zostaje rozmyty do kategorii akomodacyjnej. Pamiętaj, że członkostwa są również prezentowane w każdej kategorii. Ta sama wartość jest wprowadzana ponownie i tym razem zostaje rozmyta do bardzo akomodacyjnej kategorii. Przy ostatnim wpisie danych 12,5 widać, że tylko bardzo wąska kategoria zawiera członkostwo dla tej wartości. We wszystkich przypadkach zauważysz, że suma członkostwa wynosi 1,0.

```
fuzzfier

Please type in a category name, e.g. Cool

Enter one word without spaces

When you are done, type `done' :

v.accommodative

Type in the lowval, midval and highval

for each category, separated by spaces

e.g. 1.0 3.0 5.0 :
```

0 3 6

Please type in a category name, e.g. Cool

Enter one word without spaces

When you are done, type `done' :

accommodative

Type in the lowval, midval and highval

for each category, separated by spaces

e.g. 1.0 3.0 5.0 :

3 6 9

Please type in a category name, e.g. Cool

Enter one word without spaces

When you are done, type `done' :

tight

Type in the lowval, midval and highval

for each category, separated by spaces

e.g. 1.0 3.0 5.0 :

5 8.5 12

Please type in a category name, e.g. Cool

Enter one word without spaces

When you are done, type `done' :

v.tight

Type in the lowval, midval and highval

for each category, separated by spaces

e.g. 1.0 3.0 5.0 :

10 12 14

Please type in a category name, e.g. Cool

Enter one word without spaces

When you are done, type `done' :

done

==========

==Fuzzifier is ready for data==

==========

input a data value, type 0 to terminate

4.0

Output fuzzy category is ==> accommodative<==

category membership

---

v.accommodative 0.666667

accommodative 0.333333

tight 0

v.tight 0

input a data value, type 0 to terminate

4.0

Output fuzzy category is ==> v.accommodative<==

category membership

---

v.accommodative 0.666667

accommodative 0.333333

tight 0

v.tight 0

input a data value, type 0 to terminate

7.5

Output fuzzy category is ==> accommodative<==

category membership

---

v.accommodative 0

accommodative 0.411765

tight 0.588235

v.tight 0

input a data value, type 0 to terminate

11.0

Output fuzzy category is ==> tight<==

category membership

---

v.accommodative 0

accommodative 0

tight 0.363636

v.tight 0.636364

input a data value, type 0 to terminate

12.5

Output fuzzy category is ==> v.tight<==

category membership

---

v.accommodative 0

accommodative 0

tight 0

v.tight 1

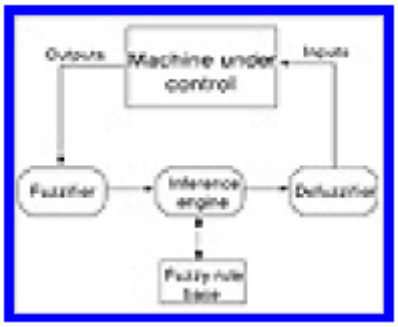
input a data value, type 0 to terminate

0

All done. Have a fuzzy day !

Rozmyte systemy sterowania

Najbardziej rozpowszechnione obecnie zastosowanie logiki rozmytej dotyczy aplikacji sterowania rozmytego. Możesz użyć logiki rozmytej, aby klimatyzator schłodził pomieszczenie. Możesz też zaprojektować system metra, który będzie wykorzystywał logikę rozmytą do sterowania układem hamulcowym w celu płynnego i dokładnego zatrzymania. System sterowania to system z zamkniętą pętlą, który zazwyczaj steruje maszyną w celu uzyskania określonej pożądanej reakcji, biorąc pod uwagę szereg danych wejściowych z otoczenia. Rozmyty system sterowania to system z zamkniętą pętlą, który wykorzystuje proces fuzzyfikacji, jak pokazano w przykładzie programu polityki Rezerwy Federalnej, do generowania rozmytych danych wejściowych do silnika wnioskowania, który jest bazą wiedzy o działaniach, które należy podjąć. Proces odwrotny, zwany rozmyciem, jest również używany w systemie sterowania rozmytego do tworzenia wyraźnych, rzeczywistych wartości, które można zastosować w maszynie lub procesie pod kontrolą. W Japonii kontrolery rozmyte są używane do sterowania wieloma maszynami, w tym pralkami i kamerami. Rysunek.3 przedstawia schemat rozmytego systemu sterowania. Główne części tego systemu zamkniętej pętli to:



• maszyna pod kontrolą - jest to maszyna lub proces, którym sterujesz np. pralka

• wyjścia - są to zmierzone zachowania odpowiedzi urządzenia, na przykład temperatura wody

• wyjścia rozmyte - są to te same wyjścia przepuszczane przez fuzzifier, np. gorące lub bardzo zimne

• silnik wnioskowania/baza reguł rozmytych - silnik wnioskowania konwertuje dane wyjściowe rozmyte na akcje, które należy wykonać, uzyskując dostęp do reguł rozmytych w bazie reguł rozmytych. Przykład reguły rozmytej: JEŚLI wyjście jest bardzo zimne, TO zwiększ nastawę temperatury wody o bardzo dużą wartość

• wejścia rozmyte - są to działania rozmyte do wykonania, takie jak zwiększenie ustawienia temperatury wody o bardzo dużą wartość

• wejścia - są to (ostre) pokrętła na maszynie do sterowania jej zachowaniem, np. ustawienie temperatury wody = 3,423, przeliczone z wejść rozmytych za pomocą defuzzifier

Kluczem do opracowania rozmytego systemu sterowania jest iteracyjne skonstruowanie bazy reguł rozmytych, które dadzą żądaną odpowiedź z maszyny. Konstruujesz te rozmyte reguły na podstawie wiedzy o problemie. W wielu przypadkach jest to bardzo intuicyjne i zapewnia solidny system sterowania w bardzo krótkim czasie.

Rozmycie w sieciach neuronowych

Rozmycie może wchodzić do sieci neuronowych w celu zdefiniowania wag ze zbiorów rozmytych. Porównanie systemów eksperckich z systemami rozmytymi jest ważne do zrozumienia w kontekście sieci neuronowych. Systemy eksperckie opierają się na ścisłych zasadach. Takie rygorystyczne zasady mogą nie zawsze być dostępne. Systemy eksperckie muszą uwzględniać wyczerpujący zestaw możliwości. Takie zbiory mogą nie być wcześniej znane. Kiedy sztywne zasady nie są możliwe i kiedy nie wiadomo, czy możliwości są wyczerpujące, podejście systemów eksperckich nie jest dobre. Niektóre sieci neuronowe, dzięki funkcjom uczenia i uczenia się, mogą funkcjonować w nieoczekiwanych sytuacjach. W tym przypadku sieci neuronowe mają przewagę nad systemami ekspertowymi i mogą zarządzać znacznie mniejszą ilością informacji, niż potrzebują systemy ekspertowe. Jedną z form rozmycia w sieciach neuronowych jest rozmyta mapa poznawcza. Rozmyta mapa poznawcza jest jak dynamiczna maszyna stanów ze stanami rozmytymi. Tradycyjna maszyna stanów to maszyna ze zdefiniowanymi stanami i wyjściami powiązanymi z każdym stanem. Przejścia ze stanu do stanu odbywają się zgodnie ze zdarzeniami wejściowymi lub bodźcami. Rozmyta mapa poznawcza wygląda jak maszyna stanów, ale ma stany rozmyte (nie tylko 1 lub 0). Na każdej ścieżce przejścia masz zestaw wag, które można nauczyć się z zestawu danych treningowych. Nasze podejście do rozmycia w sieciach neuronowych opiera się na omówieniu rozmytej pamięci skojarzeniowej, w skrócie FAM, która podobnie jak rozmyta mapa poznawcza została opracowana przez Barta Kosko.

Wyszkolone neuronowo systemy rozmyte

Do tej pory rozważaliśmy rolę logiki rozmytej w sieciach neuronowych. Odwrotna zależność, sieci neuronowe w systemach rozmytych, jest również aktywnym obszarem badań. Aby zbudować system rozmyty, musisz mieć zestaw reguł członkostwa dla kategorii rozmytych. Czasami trudno jest wydedukować te reguły członkostwa przy danym zestawie złożonych danych. Dlaczego nie wykorzystać sieci neuronowej do zdefiniowania reguł rozmytych? Sieć neuronowa jest dobra w odkrywaniu relacji i wzorców w danych i może być używany do wstępnego przetwarzania danych w systemie rozmytym. Co więcej, sieć neuronowa, która może uczyć się nowych relacji z nowymi danymi wejściowymi, może zostać wykorzystana do udoskonalenia reguł rozmytych w celu stworzenia rozmytego systemu adaptacyjnego. Trening neuronowego systemu rozmytego są używane w wielu zastosowaniach komercyjnych, zwłaszcza w Japonii:

• Laboratorium Międzynarodowych Badań Inżynierii Rozmytej (LIFE) w Jokohamie w Japonii posiada sieć neuronową propagacji wstecznej, która wyprowadza reguły rozmyte i funkcje członkostwa. System LIFE został z powodzeniem zastosowany do systemu wsparcia handlu dewizowego z około 5000 regułami rozmytymi.

• Firma Ford Motor Company opracowała możliwe do przeszkolenia systemy rozmyte do kontroli prędkości jazdy na biegu jałowym w samochodach.

• National Semiconductor Corporation posiada oprogramowanie o nazwie NeuFuz, które obsługuje generowanie reguł rozmytych za pomocą sieci neuronowej dla aplikacji sterujących.

• Wiele japońskich produktów konsumenckich i przemysłowych wykorzystuje sieci neuronowe z systemami rozmytymi, w tym odkurzacze, urządzenia do gotowania ryżu, pralki i kserokopiarki.

• Firma AEG Corporation z Niemiec używa w swojej pralce oszczędzającej wodę i energię wyszkolonego przez sieć neuronową systemu sterowania rozmytego. Po załadowaniu pralki mierzy poziom wody za pomocą czujnika ciśnienia i określa ilość prania w pralce na podstawie prędkości i objętości wody. W sumie 157 reguł zostało wygenerowanych przez sieć neuronową, która została wytrenowana na danych korelujących ilość prania z pomiarem poziomu wody na czujniku.

Podsumowanie

Tu przeczytasz o logice rozmytej, zbiorach rozmytych i prostych operacjach na zbiorach rozmytych. Logika rozmyta, w przeciwieństwie do logiki Boole'a, ma więcej niż dwie kategorie włączenia lub wyłączenia do opisania zachowania systemów. Używasz wartości członkostwa dla danych w kategoriach rozmytych, które mogą się pokrywać. W tym rozdziale opracowałeś również program fuzzifier w C++, który pobiera wartości ostre i konwertuje je na wartości rozmyte, na podstawie zdefiniowanych przez Ciebie kategorii i przynależności. W przypadku sieci neuronowych logika rozmyta może służyć jako filtr przetwarzania końcowego lub przetwarzania wstępnego. Kosko opracował sieci neuronowe wykorzystujące rozmycie i nazwał je rozmytą pamięcią skojarzeniową, co zostanie omówione w dalszych rozdziałach. Przeczytałeś również o tym, jak sieci neuronowe mogą być używane w systemach rozmytych do definiowania funkcji przynależności i reguł rozmytych.